UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/712,384	11/12/2003	William John Gallagher	BEAS-01316US3	9603

23910 7590 06/01/2007

FLIESLER MEYER LLP
650 CALIFORNIA STREET
14TH FLOOR
SAN FRANCISCO, CA 94108

EXAMINER	
NGUYEN, PHILLIP H	
ART UNIT	**PAPER NUMBER**
2191	

MAIL DATE	DELIVERY MODE
06/01/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/712,384	GALLAGHER, WILLIAM JOHN	
	Examiner	Art Unit	
	Phillip H. Nguyen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 April 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-25* is/are pending in the application.

4a) Of the above claim(s) *22-24* is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-21 and 25* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *20070330*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed on 4/19/2007.

2. Per Applicant's request:

- Claims 22-24 have been canceled. Claim 1 has been amended. Claim 25 is newly added.
- Claims 1-21 and 25 are pending and have been considered below.

Note:

3. Regarding claim 1 recites the word "configured" in the body of the claim. It indicates intended use and as such does not carry patentable weight. The limitation following the phrase "configured" describes only intended use but not necessarily required functionality of the claim.

Specification

4. The use of the trademark JAVA has been noted in this application (the specification). It should be capitalized wherever it appears and be accompanied by the generic terminology.

Although the use of trademarks is permissible in patent applications (the specification), the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner, which might adversely affect their validity as trademarks.

Claim Rejections - 35 USC § 112

5. The amendment filed on 3/30/2007 overcomes the rejection set forth to claim 22 of previous action. Therefore, the rejection is withdrawn.

- Regarding new claim 25 contains the trademark or trade name JDBC and EJBs. Where a trademark or trade name is used in a claim as a limitation to identify or describe a particular material or product, the claim does not comply with the requirements of 35 U.S.C. 112, second paragraph. See Ex parte Simpson, 218 USPQ 1020 (Bd. App. 1982). The claim scope is uncertain since the trademark or trade name cannot be used properly to identify an particular material or product. A trademark or trade name is used to identify a source of goods, and not the goods themselves. Thus, a trademark or trade name does not identify or describe the goods associated with the trademark or trade name. In the present case, the trademark or trade name is used to identify or describe a family of products generated in the proprietary programming language called JAVA and accordingly, the identification or description is indefinite.

Response to Arguments

6. Applicant's arguments filed 3/30/2007 have been fully considered but they are not deemed persuasive.

Applicant asserts on page 8-9 of the amendment that Boehm does not teach "dynamically generated code would be configured to exist for the life of the server it resides upon." However, Boehm teaches "the classes and objects need only exist at the time that a running application calls for the adapter classes, and can be dynamically modified or exchanged in order to optimize the running application or modify application functionality." (col. 3, lines 4-8).

Examiner respectfully disagrees with all the allegations as argued. One of ordinary skill in the art would have been recognized that for the purpose of dynamic code generation, the

classes and objects are only created during runtime (dynamic) and will be last until they fulfill the purpose of dynamic code generation. Boehm does not say that the classes and objects will be deleted before the life of a server ended. Even assuming the classes and objects will be deleted before the life of the server ended, they can be configured to exist for the life of the server to fulfill the purpose of dynamic code generation.

Examiner is entitled to give claim limitations their broadest reasonable interpretation in light of the specification. See MPEP 2111 [R-1] Interpretation of Claims-Broadest Reasonable Interpretation. During patent examination, the pending claims must be 'given the broadest reasonable interpretation consistent with the specification.' Application always has the opportunity to amend the claims during the prosecution and broad interpretation by the examiner reduces the possibility that the claims, once issued, will be interpreted more broadly than is justified. In re Prater, 162 USPQ 541, 550-51 (CCPA 1969).

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

8. Claims 1-10, 12, 14-18 and 25 are rejected under 35 U.S.C. 102(e) as being anticipated by Boehme et al. (United States Patent No.: US 6,578,191 B1).

As per claim 1:

Boehme discloses:

- creating a class file container object ("**Adapter classes and objects are automatically and dynamically generated**" col. 3, line 2-3; "**ClassInfo newClass=new ClassInfor** ..." col. 4, line 30-33);
- adding a method to the class file object (see at least col. 4, line 42 "**newClass.addMethod(ACC_PUBLIC, "actionPerformed", "(LactionEvent;)V", bytecodes)**);
- adding code to the method using programming language constructs ("**code is created for the adapter class initialization methods**" col. 5, line 49-50; "**code is created for the adapter class methods referenced in the adapter class specification**" col. 5, line 52-53);
- generating byte code for the class file container object ("**bytecodes necessary to construct the adapter class, interface, field, methods, and attributes are generated**" col. 4, line 24-26);
- instantiating an instance of the new class file object ("**An instance of the adapter class is instantiated in function block 107**" col. 4, line 56); and
- generating executable code from the bytecode by using a class loader (see at least col. 4, line 48-52 "**the adapter class is then loaded into the running application...Loader ldr=new BMLLoader();...**", also see at least FIG. 1, item 105).

- wherein dynamically generated code (see at least col. 3, line 2-3 "**Adapter classes and objects are automatically and dynamically generated**...") would be configured to (intended use) exist for the life of the server it resides upon.

As per claim 2:

Boehme discloses:

- setting attributes for a class file ("**newClass.setSuperClassName("com/ibm/bml/EventAdapterImpl")**;" col. 4, line 35, setting the super class name).

As per claim 3:

Boehme discloses:

- a class file name ("**newClass.setSourceFilename("BMLActionEventAdapter")**" col. 4, line 32); and
- a parent super class ("**newClass.setSuperClassName("com/ibm/bml/EventAdapterImpl")**;" col. 4, line 35).

As per claim 4:

Boehme discloses:

- adding a plurality of methods to the class file object ("**the bytecodes necessary to construct... methods**" col. 4, lines 24-25; **newClass.addSpecialMethod(...); newClass.addMethod(...);**" col. 4, lines 40-43).

As per claim 5:

Boehme discloses:

- the programming language constructs correspond to programming language statements, expressions, and variables (col. 4, lines 10-20, **this method contains statements, expressions, and variables**...).

As per claim 6:

Boehme discloses:

- parameters ("**beanInfo=Introspector.getBeanInfo(jugglerClass)...**" col. 4, line 10, **jugglerClass is a parameter**).

As per claim 7:

Boehme discloses:

- wherein each statement, expression type, and variable is represented as an object (for example, **"<bean class = "java.awt.button">** col. 3, line 62, this is a statement contains an expression of bean class set equal to java.awt.button object;

"beanInfo=Introspector.getBeanInfo(jugglerClass)" col. 4, line 10, jugglerClass is a variable and represents an object).

As per claim 8:

Boehme discloses:

- generating an intermediate representation of program flow ("**op-codes**" col. 4, line 27).

As per claim 9:

Boehme discloses:

- converting the intermediate representation into byte code ("**the bytecodes necessary to construct the... are generated based upon the wiring description, and the op-codes**" col. 4, lines 24-28).

As per claim 10:

Boehme discloses:

- wherein the program code implements an adapter class ("**to construct adapter class**" col. 4, lines 24-25; also see col. 4, lines 30-55).

As per claim 12:

Boehme discloses:

- repeatedly adding a method to the class file object for each method associated with a stub generated for a remote object (col. 4, lines 40-43, **it repeatedly adds methods to the class**).

As per claim 14:

Boehme discloses:

- repeatedly adding code for each method added to the class file ("**code is created for the adapter class initialization methods**" col. 5, lines 49-50; "**code is created for the adapter class methods referenced in the adapter class...**" col. 5, lines 52-53, **which means, code is created repeatedly for each method in order to fulfill the functionality of the methods**).

As per claim 15:

Boehme discloses:

- generating a tree of statements (col. 4, lines 10-20, **a tree represents at least one method containing at least one code statement, expression, variable, or other programming construct**).

As per claim 16:

Boehme discloses:

- generating a tree representing at least one method (col. 4, lines 10-20, **is a method**), the at least one method comprising at least of code statement, an expression, a variable and a

programming construct (col. 4, lines 10-20, **this method contains statements, expressions, and variables**).

As per claim 17:

Boehme discloses:

- generating a tree forming a known structure when the class file container is a known type ("**adapter type**" col. 3, line 57).

As per claim 18:

Boehme discloses:

- generating a tree forming a known structure when the class file container is of at least one of an adapter and a proxy type ("**adapter type**" col. 3, line 57).

As per claim 25:

Boehme discloses:

- wherein the dynamically gernerated code is used for remote method invocaition skeletons, remote method invocation stubs, remote method invocation stubs (**remote method invocation is a mechanism that is part of the JAVA program language. It allows JAVA objects to invoke methods on objects from another JVM. Therefore, it is inherent in Boehme's approach in order to invoke methods for dynamically creating code)**, wrapper for JDBC connections, and proxies used to enforce call-by-value semantics between EJBs.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. Claims 1-10, 12, 14-18 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Boehme et al. (United States Patent No.: US 6,578,191 B1).

As per claim 1:

Boehme discloses:

- computer code for creating a class file container object ("**Adapter classes and objects are automatically and dynamically generated**" col. 3, lines 2-3; "**ClassInfo newClass=new ClassInfor** ..." col. 4, lines 30-33);
- computer code for adding a method to the class file object (see at least col. 4, line 42 "**newClass.addMethod(ACC_PUBLIC, "actionPerformed", "(LactionEvent;)V", bytecodes)**);
- computer code for adding code to the method using programming language constructs ("**code is created for the adapter class initialization methods**" col. 5, lines 49-50; "**code is created for the adapter class methods referenced in the adapter class specification**" col. 5, lines 52-53);

- computer code for generating byte code for the class file container object ("**bytecodes necessary to construct the adapter class, interface, field, methods, and attributes are generated**" col. 4, lines 24-26);
- computer code for instantiating an instance of the new class file object ("**An instance of the adapter class is instantiated in function block 107**" col. 4, line 56); and
- computer code for generating executable code from the bytecode by using a class loader (see at least col. 4, lines 48-52 "**the adapter class is then loaded into the running application...Loader Idr=new BMLLoader();...**").

Boehme does not explicitly disclose:

- wherein dynamically generated code would be configured to (intended use) exist for the life of the server it resides upon.

However, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to recognize that the generated code in Boehme's approach would be configured to exist for the life of a server it resides upon if one wants to keep it for longer period of time. Therefore, One would have been motivated to configure the generated code in Boehme's approach to exist for the life of a server it resides upon if needed in order to fulfill the purpose of code generation.

As per claim 2:

Boehme discloses:

- setting attributes for a class file ("**newClass.setSuperClassName("com/ibm/bml/EventAdapterImpl")**;" col. 4, line 35, setting the super class name).

As per claim 3:

Boehme discloses:

- a class file name ("**newClass.setSourceFilename("BMLActionEventAdapter")**" col. 4, line 32); and
- a parent super class ("**newClass.setSuperClassName("com/ibm/bml/EventAdapterImpl")**;" col. 4, line 35).

As per claim 4:

Boehme discloses:

- adding a plurality of methods to the class file object ("**the bytecodes necessary to construct... methods**" col. 4, lines 24-25; **newClass.addSpecialMethod(...); newClass.addMethod(...);**" col. 4, lines 40-43).

As per claim 5:

Boehme discloses:

- the programming language constructs correspond to programming language statements, expressions, and variables (col. 4, lines 10-20, **this method contains statements, expressions, and variables**...).

As per claim 6:

Boehme discloses:

- parameters ("**beanInfo=Introspector.getBeanInfo(jugglerClass)...**" col. 4, line 10, **jugglerClass is a parameter**).

As per claim 7:

Boehme discloses:

- wherein each statement, expression type, and variable is represented as an object (for example, "**<bean class = "java.awt.button">** col. 3, line 62, this is a statement contains an expression of bean class set equal to java.awt.button object; "**beanInfo=Introspector.getBeanInfo(jugglerClass)**" col. 4, line 10, jugglerClass is a variable and represents an object).

As per claim 8:

Boehme discloses:

- generating an intermediate representation of program flow ("**op-codes**" col. 4, line 27).

As per claim 9:

Boehme discloses:

- converting the intermediate representation into byte code ("**the bytecodes necessary to construct the... are generated based upon the wiring description, and the op-codes**" col. 4, lines 24-28).

As per claim 10:

Boehme discloses:

- wherein the program code implements an adapter class ("**to construct adapter class**" col. 4, lines 24-25; also see col. 4, lines 30-55).

As per claim 12:

Boehme discloses:

- repeatedly adding a method to the class file object for each method associated with a stub generated for a remote object (col. 4, lines 40-43, **it repeatedly adds methods to the class**).

As per claim 14:

Boehme discloses:

- repeatedly adding code for each method added to the class file ("**code is created for the adapter class initialization methods**" col. 5, lines 49-50; "**code is created for the adapter class methods referenced in the adapter class...**" col. 5, lines 52-53, **which**

means, code is created repeatedly for each method in order to fulfill the functionality of the methods).

As per claim 15:

Boehme discloses:

- generating a tree of statements (col. 4, lines 10-20, **a tree represents at least one method containing at least one code statement, expression, variable, or other programming construct**).

As per claim 16:

Boehme discloses:

- generating a tree representing at least one method (col. 4, lines 10-20, **is a method**), the at least one method comprising at least of code statement, an expression, a variable and a programming construct (col. 4, lines 10-20, **this method contains statements, expressions, and variables**).

As per claim 17:

Boehme discloses:

- generating a tree forming a known structure when the class file container is a known type ("**adapter type**" col. 3, line 57).

As per claim 18:

Boehme discloses:

- generating a tree forming a known structure when the class file container is of at least one of an adapter and a proxy type ("**adapter type**" col. 3, line 57).

As per claim 25:

Boehme discloses:

- wherein the dynamically gernerated code is used for remote method invocaition skeletons, remote method invocation stubs, remote method invocation stubs (**remote method invocation is a mechanism that is part of the JAVA program language. It allows JAVA objects to invoke methods on objects from another JVM. Therefore, it is inherent in Boehme's approach in order to invoke methods for dynamically creating code)**, wrapper for JDBC connections, and proxies used to enforce call-by-value semantics between EJBs.

11. Claims 11 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Boehme et al. (United States Patent No.: US 6,578,191 B1), in view of Cohen et al. (United States Patent No.: 6,011,918).

As per claim 11:

Boehme does not explicitly disclose:

- wherein the program code implements a proxy class.

However, Cohen discloses an analogous computer program product wherein the program code implements a proxy class ("**proxy class is called X, ... will execute on the local machine**" col. 15, lines 32-34). Therefore, it would have been obvious to one having an ordinary skill in the art to modify Boehme's approach to implement proxy class. One having an ordinary skill in the art would have been motivated to implement proxy class so that **each of the constructed methods make corresponding remote method calls using Java's RMI function** (see Cohen col. 14, lines 5-10).

As per claim 13:

Boehme does not explicitly disclose

- wherein the program code for repeatedly adding a method to the class file object for each method associated with a stub generated for a remote object includes program code for: determining a number of methods associated with the stub in a remote interface.

However, Cohen discloses an analogous computer program product for determining a number of calling methods ("**the weighting of the relationships between the programmed methods is performed by determining for each calling programmed method**" col. 3, lines 52-55). Therefore, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to modify Boehme's approach to determine the number of method added to the class file object. One having an ordinary skill in the art would have been motivated **to determine the number of method by include a counter or an indicator that increment each time a method is adding to the class file object so it would know when it reaches the end of the adding iteration process.**

12. Claims 19-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Boehme et al. (United States Patent No.: US 6,578,191 B1), in view of Stapp et al. (United States Patent Application Publication No.: US 2004/0015832 A1).

As per claim 19:

Boehme does not explicitly disclose:

- wherein the program code for generating byte code for the class file container object includes program code for maintaining a state of a program being generated by each statement.

However, Stapp discloses an analogous computer program product that includes program code for maintaining a state of a program being generated by each statement ("**Beans also have persistence, which is a mechanism for storing the state of a component in a safe place**" Paragraph 0039). Therefore, it would have been obvious to one having an ordinary skill in the art to modify Boehme's approach to maintaining a state of a program being generated. One having an ordinary skill in the art would have been motivated to modify Boehme's approach **because it allows a component (bean) to retrieve data that a particular user had already entered in an earlier user session** (Paragraph 0039).

As per claim 20:

Stapp discloses:

- maintaining at least one of a content of a stack, a content of a variable in use during program flow ("**a component (bean) to retrieve data that a particular user had**

already entered in an earlier user session" Paragraph 0039, **data is contents of variables**).

As per claim 21:

Boehme discloses:

- generating an intermediate representation of program flow based upon the at least one of a content of a stack, a content of a variable in use during program flow ("**the bytecodes necessary to construct the adapter class, interface, fields, methods, and attributes are generated**..." col. 4, lines 24-29, **op-codes are intermediate representation of program flow can be found in bytecode, specifies operation to perform**).

Conclusion

13. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Foti (United States Patent No.: US 7,181,745 B1) discloses a method and system for accessing objects defined within an external object-oriented environment.
- Jones et al. (United States Patent No.: US 6,877,163 B1) discloses a method and system for dynamic proxy classes.
- Guthrie et al. (United States Patent No.: 6,549,955 B2) discloses a method and system for dynamic generation of remote proxies.

14. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Phillip H. Nguyen whose telephone number is (571) 270-1070. The examiner can normally be reached on Monday - Thursday 10:00 AM - 3:00 PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y. Zhen can be reached on (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

PN
5/21/2007

WEI ZHEN
SUPERVISORY PATENT EXAMINER